Prototyping of creation, implementation and visualization of correlation rules and microDocs

Alexandra Stenzel[1*], and *Florian* Rommel[1**]

[1]Karlsruhe University of Applied Sciences, Faculty of Information Management and Media, 76133 Karlsruhe, Germany

Abstract. Semantic Correlation Rules (SCR) and microDocs are new concepts in the field of content delivery. SCR allow to define relationships between information units based on their metadata and, therefore, allow for the dynamic aggregation of microDocs. The creation of SCR heavily relies on the capabilities of modern content management systems (CMS) or ontology editors. The evaluation and visualization of the emerging microDocs, on the other hand, relies on the capabilities of content delivery portals (CDP). At this time, the support of both concepts in most software solutions currently being used, is only partly existent. This paper aims to demonstrate, how these currently existing limitations can be overcome, to reveal important factors to be considered and to showcase future possibilities of the aforementioned concepts. For this purpose, we developed a series of prototypes and conceptional visuals regarding creation of SCR, aggregation of microDocs and their visual appearance taking human perception into account.

1 Introduction

microDocs are a new concept in the field of content delivery. On one hand, they aim to bridge the gap between large documents providing too much information. On the other hand, they aim at topics providing too little context for the users' specific information need. microDocs can be considered as an aggregated (sub-)set of topics that are correlated in the scope of the users' particular use case [1].

The rules by which microDocs are being aggregated are called Semantic Correlation Rules (SCR). SCR do not address certain topics directly. Instead, they act a higher level and interconnect topics by their classification. According to Ziegler [1], there are different implementation levels with increasing complexity for the creation of SCR, reaching from static manual approaches to artificial intelligent based approaches. In this paper, we will focus on a manual software-supported approach to create SCR.

Once the SCR are created and the resulting microDocs are aggregated, the question on how microDocs are being presented to the user remains unanswered. There are many possible solutions to be considered. These range from simple solutions (list of links) to topic excerpts, topic/term clouds or tree map like visualizations [1]. In this paper, we will explore different visualization possibilities with particular focus on the context of human perception and design rules [2].

As microDocs are a new concept in the field of content delivery, the software-support of current content delivery portals (CDP) is only partly existent. In order to be able to test the aggregation results and, moreover, to practically explore different ways to display the generated microDocs, we developed a prototype of a CDP, utilizing an existing iiRDS-API acting as a testbed for current and upcoming studies on the aforementioned concepts.

Thus, the primary aim of this paper is to show how SCR can be created, to explore how the resulting microDocs can be presented to the user, and also to create an interactive and highly customizable testbed for our studies.

This paper is structured as follows: In section 2, we propose an alternative method for the creation of Sematic Correlation Rules. Section 3 is focusing on design criteria and derives proposals for the visual representation of microDocs. In section 4 we will outline the development of a CDP prototype, which acts as a front-end for an iiRDS-based API. Finally, in section 5, we will discuss study limitations and in section 6, will provide conclusions as well as an outlook.

* Alexandra Stenzel: stal1035@hs-karlsruhe.de
** Florian Rommel: hska@florianrommel.de

2 Prototyping of SCR Creation

The creation of SCR currently depends on the content management system (CMS) the user works with and to what extent this CMS supports the function of SCR. Only a few CMS support the creation of SCR. For this reason, it is necessary, to find a solution that allows the user to create a SCR independent of CCMS. We address the question of how it may become possible for anyone who works with metadata to create SCR. In order to be independent of the type of software the user owns and to be platform independent, we decided to outline such a web-based SCR creation tool.

The tool should be able to fulfill the following requirements:

- RDF files should be able to be uploaded and read.
- Metadata should be able to be taken from the RDF files in their hierarchy.
- SCR creation should be supported and made possible by a graphical user interface.
- SCR creation should be easy to understand.
- The SCR should be written to an RDF file in the background.
- For an import into any CMS, the RDF file should be exportable.

2.1 Principles of user experience design, human perception, and their impact on the design of the prototype

In addition to the functions relating to the SCR concept, the tool should also be designed in such a way, that the basic rules for proper usability and a positive user experience are observed [3]. Basically, usability is higher when the design corresponds to the generally accepted conventions and to the user's expectations. Focus should be on the usual operating concepts of common software and modern websites. During the design of the prototype, care was taken to ensure, that the user was always aware of the current stage of the creation process. Orientation is supported with the help of headings or the arrangement of elements on the page.

To increase the acceptance of the tool, it is advised, to give the user as much control of the tool as possible [3]. To satisfy this need, for example, buttons were included to undo steps. Furthermore, this is necessary to offer a way to the user to save his work status at any time and to be able to continue, where he had previously stopped. Consequently, it should be possible to create an RDF file at any time. At first glance, one or two functions seem pointless for the creation process. However, in order to increase the user's acceptance, it must be possible to experiment with the tool as freely as possible. If errors take place during the creation process, error messages appear that clearly communicate, what the error was about and how it can be fixed. To further increase the understanding of the tool, all work steps, that require explanation, are explained in the appropriate context.

The individual elements of the tool are arranged according to the principles on the basis of which we form element groups or perceive the difference of elements. For example, the rules of a use case are always delimited from the following or previous use case with two lines. Likewise, all InRules are the same color, while the OutRules are a different color in order to separate them from the InRules.

To find out, whether the design and organization of the elements were successful, a usability test should be performed before development.

2.2 Proposal for realization

The result of the proceeding, regarding conceptual considerations and design-relevant principles, is a graphical and minimally functional prototype. In the process of developing the final and functional tool, this represents the first approach. The starting page offers a short introduction to the subject of SCR. On this page, the user can upload his RDF file with the included metadata. If the data has been uploaded successfully, the user receives feedback on the interface as shown in figure 1.

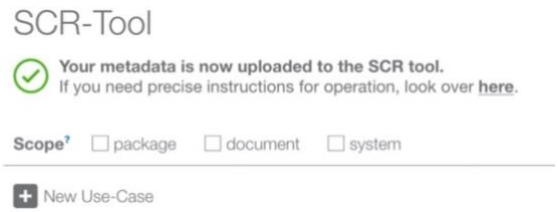

Fig. 1. Feedback after uploading the RDF file

The tool is structured in such a way, that it starts by defining a use case. This use case then has one InRule and several OutRules. Once the use case is defined, the use case specific InRule can be created. The uploaded metadata is displayed in a drop-down list and a metadatum can be selected. Each metadata can be edited or deleted during the definition of the InRule. This is a feature mentioned above to increase usability. Once all metadata of the InRule are entered, it can be closed by clicking the button "Save InRule". Also, in this case, the InRule can be edited or deleted during the entire creation process.

The OutRules offer more setting options than the InRule. Now, there is an additional option for each metadata to specify the type of relationship. The options are "equals" and "selects". If the user does not select either relationship type, the code defaults to the "hasCorrelation" relationship type. Once the metadata of an OutRule is defined, the user can specify the strength of the relationship between this OutRule and the InRule. All of the settings the user enters via the graphical user interface are defined in the code as shown in figure 3.

```
<scr:OutRule rdf:about="http://www.i4icm.de/scr#OutRule1_1">
  <rdf:type rdf:resource="http://www.i4icm.de/scr#OutRule"/>
  <rdfs:label xml:lang="en">OutRule1_1</rdfs:label>
  <scr:selects rdf:resource="http://www.hs-karlsruhe.de/sim#manual"/>
  <scr:selects rdf:resource="http://www.hs-karlsruhe.de/sim#controller"/>
  <scr:selects rdf:resource="http://www.hs-karlsruhe.de/sim#Bosch_Smart_Home"/>
  <scr:equals rdf:resource="http://www.hs-karlsruhe.de/sim#smart_home_system"/>
  <scr:selects rdf:resource="http://www.hs-karlsruhe.de/sim#climate_monitoring"/>
  <scr:Strength rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">8</scr:Strength>
</scr:OutRule>
```

Fig. 2. The exemplary OutRule as code

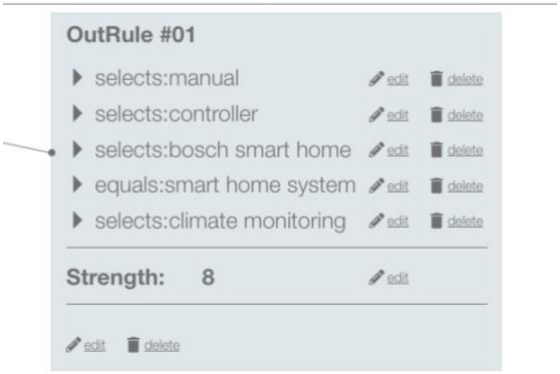

Fig. 3. Exemplary OutRule

3 Visualization of MicroDocs

In order to visualize MicroDocs, it must first be established, what exactly is to be achieved by the visualization. The visualization must be able to clearly show, how close or far away a topic is from another topic in terms of use case. Furthermore, it has to be considered, how much information about a referenced topic should be shown in the view.

3.1 Design criteria

Humans subconsciously derive the importance of objects from how they are positioned, colored or how big or small they are. That is why in this case, it is necessary, to work with the parameters of size, position and colorfulness [10]. The bigger an object is or the stronger the color is, the more important this object appears to the viewer. If the color is weaker, the object will lose the importance for the viewer. The position should be chosen, so that the object does not appear to be too important or too unimportant. Here, attention should be paid to the conventions that have been established successfully. However, in this case, the object should not be positioned too prominently, otherwise it would compete with the currently displayed topic in its visually perceived importance.

3.2 Possible implementations

The first result are simple rectangles that appear in the main area of the topic window below the current content. As shown in figure 4, the position is thus subordinate to the current topic, but important enough to be perceived as relevant information. The individual referenced topics are displayed as colored squares of different sizes.

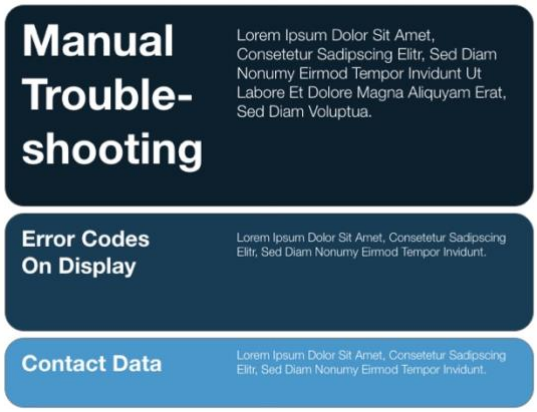

Fig. 4. Possible visualization of the MicroDocs by colored rectangles

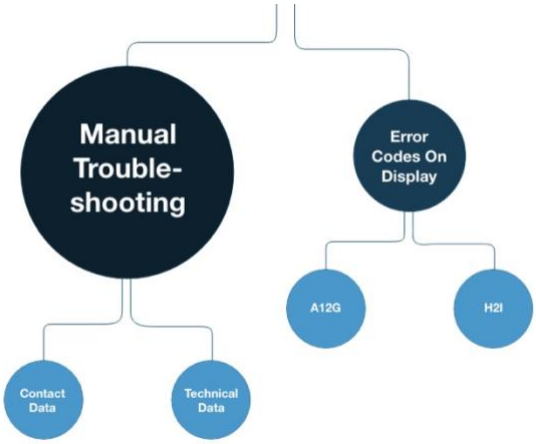

Fig. 5. Possible visualization of the MicroDocs by colored circles

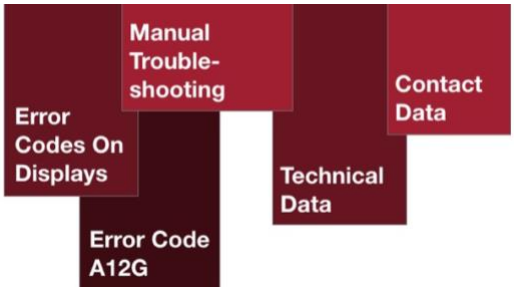

Fig. 6. Visualization of the MicroDocs by colored bars

As in the previous visualization, the second result shows the correlating topics in different sizes. The size is again an indicator for the relevance to the main topic. What is added here, however, is the visualization of the relationships between the topics. In this visualization, not only the thematic proximity, but also the position of the topic in the common information structure are shown.

The third result relates to layers. Bars overlap and, therefore, form a hierarchy. Here, the brightness value of the colors is now perceived differently. The deeper the layer, the darker the hue and the less important the effect. No matter how the MicroDocs are presented, focus should be placed less on design in itself but more on simplicity and concreteness. It should be clear to the recipient at first glance, what the reference topics are and how they relate to the main topic.

4 Developing a CDP prototype

With the development of our CDP prototype, we pursued a number of goals. The prototype should act as an interactive testbed for studying different aspects in the field of content delivery. It should be highly extensible and customizable to serve as a software foundation for further studies. For this study, it should primarily act as a testbed for the evaluation of SCR and for testing our visual concepts for the representation of microDocs.

4.1 Technical overview

Before explaining the implementation details, we provide a brief overview of the CDP prototype we developed. The application utilizes a client-server architecture. The server runs a RESTful API and provides endpoints for data consumption. The client runs a web application that is requesting resources from the server by calling the provided endpoints.

The API we used for our prototype is utilizing iiRDS packages as input format. Thus, iiRDS packages can be exported from a content management system (CMS) or can be created by other means. Then, the packages can be uploaded to the iiRDS-based API, which in turn will provide the contents of these packages through its endpoints. The front-end application consumes these endpoints and displays the results of the requests to the user.

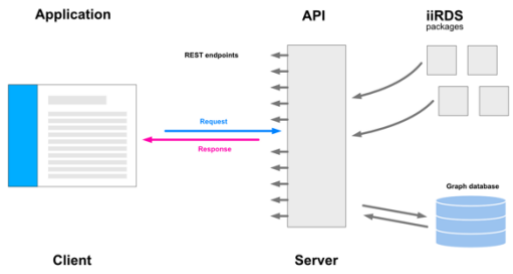

Fig. 7. Schematic representation of the implemented software architecture.

4.2 Reasons for using an iiRDS-based API

The intelligent information request and delivery standard (iiRDS) is an arising standard for the exchange of modularized technical information. It can be used to transfer metadata-enriched information units from vendor A to vendor B, or to transfer information units between systems like a CMS and a CDP, for example. Therefore, iiRDS has its own standardized classification scheme based on the PI-classification method [5]. Utilizing this new standard provides some advantages. As iiRDS is a standardized exchange format, it allows our prototype to receive classified contents from different sources. As long as the source system is able to export its contents as iiRDS packages, we can make use of these contents and deliver them throughout our CDP prototype. Furthermore, by using iiRDS as exchange format, we are able to use an already existing, proprietary iiRDS-based RESTful API developed by c-rex.net GmbH [6], that stores the information units and provides retrieval mechanisms. Utilizing an API that provides data storage and data retrieval simplifies the development significantly, as we do not need to implement these functionalities on our own. Instead, we can focus on data representation and user interactions. Since the c-rex API builds on iiRDS as input format, the endpoints derive from the iiRDS classification scheme making the API relatively easy to approach, as classification scheme and retrieval mechanisms are closely coupled. Additionally, the c-rex iiRDS API is already implementing SCR and, therefore, allows for the aggregation of microDocs.

4.3 The c-rex iiRDS API

The c-rex iiRDS API we utilized is implementing the REST paradigma [7]. To illustrate a typical transaction between the consuming application and the serving API, the following example may be considered: The application sends a request to the API, requesting all topics in the system. The URI for requesting the resources has the following form: http://localhost/iirds/v1/Topics. The API will respond to this request by sending a HTTP message including the requested information, in this particular case, a list of all topics.

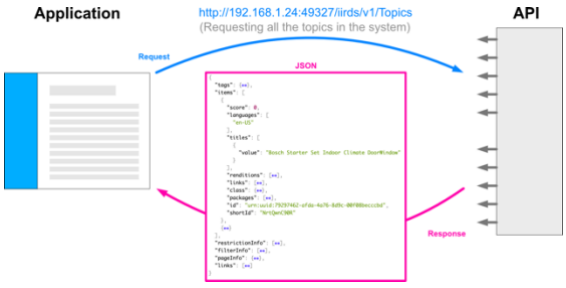

Fig. 8. Schematic representation of the transaction between the application and the iiRDS API.

4.4 The front-end application framework

For the development of the front-end application, we used an open-source, client-side JavaScript framework called Vue. Vue is designed for the development of single-page applications (SPA). The framework implements a model-view-viewmodel (MVVM) pattern [8] and supports declarative rendering allowing for a mechanism called 2-way data binding. 2-way data binding basically means, that inputs and outputs are directly bound to the data source. This eliminates the need for developers to manually interact with DOM elements to change values or receive user inputs. When the application state changes, the view will automatically get updated. Vue applications can be modularized into components, meaning, that it is possible to break up the entire applications business logic and user interface into multiple, more simple and reusable building blocks. Components can be nested and are able to pass properties between each other, so it is possible to inherit processed values from parent components to child components. Furthermore, Vue is well documented and, compared with other modern front-end frameworks, considered as easy to learn. An active developer community around this framework also exists, providing a large number of ready-to-use extensions. As the front-end application is based on web standards like HTML, CSS and JavaScript, we can also make use of a wide variety of open-source third-party libraries and frameworks to extend the functionality of our application. Altogether, these properties significantly reduced the lines of code to be written and allowed for fast prototyping of our CDP front-end application.

4.5 The CDP front-end application

The front-end application itself has to fulfil a number of tasks. It has to provide a user interface (UI), in which the user can interact with the applications business logic. It further needs to dynamically (based on users' input) interact with the API endpoints to request information units. Additionally, it needs to process the data coming in from the API response and build a view (displaying the information units). The applications UI is kept simple in design and function. For the basic style of the application, we made use of the front-end CSS framework Bootstrap (as Vue implementation) [10]. We developed a number of views to enable users to navigate through the contents stored in the API and for displaying the retrieved information units. Each view is serving a single purpose, like displaying all packages in the system, displaying a structured document with a table of contents, or filtering topics by their metadata. The view shown in figure 7 is listing topics and provides content filters.

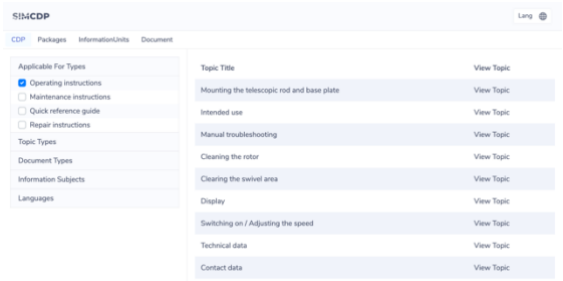

Fig. 9. Application view for filtering topics by their metadata. On the left, content filters are shown. On the right, a table containing the filtered retrieval results is shown.

When the user selects a metadata item, the application immediately will send a request to the API to deliver the filtered contents. Once the response of the API arrives, the application will update the view accordingly. The UI is also the place, where we test the visual appearance of the aggregated microDocs. The visual appearance shown in figure 10 is dynamically generated by a third-party JavaScript library for data visualizations [9]. In this case, the microDocs are displayed as a simplified tree map. The colors and size of the items derive from the correlation strength. This is only one example on how third-party libraries can be used to display microDocs.

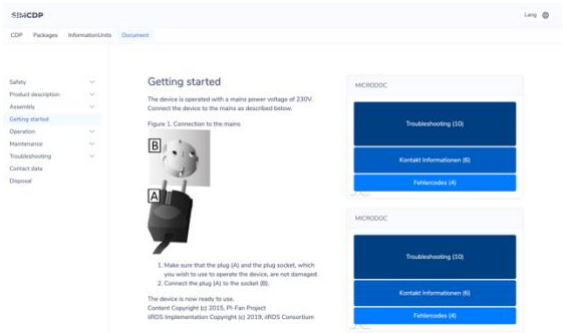

Fig. 10. Application view for displaying topics of a document. The right side is showing microDocs as a simplified tree map.

The application is built out of nested components. It has a root component holding the header, including navigation items, and an area for dynamically displaying child components based on the selected navigation item. These components consist of a template (dynamically generated HTML), code logic like the API calls and resolving functions (JavaScript), and styles (CSS). The component showing a certain topic, for example, will request the topic contents (HTML, images) from the API, it will then resolve the response, prepare the incoming data and dynamically render the view. Furthermore, it will examine, whether the topic is correlated with other topics by sending a request to the InRule endpoint. If the topic is correlated, the topic component will invoke a microDocs component, which will render the received microDocs. As mentioned before, parent components can pass

processed values to child components. This is especially useful, for example, when testing different visual styles for displaying microDocs, as shown in figure 11. The business logic responsible for retrieving the data from the API is placed in a parent component, while the logic for preparing the visual appearance is placed in a child component. To test another visual appearance, a new child component must be developed, while the retrieval logic placed in the parent component is simply being reused.

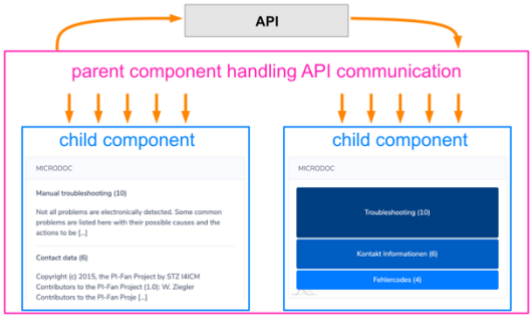

Fig. 11. Schematic representation of nested components.

5 Study limitations

During this study, we were faced with the following limitations. First, during the given time period, we were not able to fully implement all proposed visuals in the CDP prototype. Therefore, we were unable to carry out usability tests that could validate our assumptions on the representation of microDocs and the users' actual perception of those in the tool itself. During the development phase of the CDP prototype, we further made the decision to utilize iiRDS packages and SCR built and delivered by the API vendor (c-rex) for the development and to test the retrieval results. Thus, we deviated from our initial idea to use custom-built packages and SCR. In the scope of our studies – to build prototypes and explore the iiRDS standard and the concept of SCR – we found this procedure justifiable, as it would not distort the results.

6 Conclusion and outlook

During the development of the SCR creation prototype, we came to the conclusion that such a tool cannot only be used to actually create SCR, but is also able to explain the concept as such, by guiding the user through the creation process and by providing further explanation for each step towards the end result. As the given result is still a prototype focusing on the creation process and user interaction, and not a productive tool that actually can export an RDF file, the next step would be to develop a productive software solution based on the prototype. Further studies could then explore out- and input operation in different CMS and CDP platforms.

We believe the CDP prototype based on the c-rex iiRDS-API to be a promising solution for the future. In the given timeframe, were able to implement basic retrieval mechanisms, microDocs aggregation mechanisms and different representations of the latter. Further studies should be carried out, to implement more advanced features and visual representations as well as to validate the user experience.

Special thanks to the Faculty of Information Management and Media for the financial support and Markus and Chris Wiedenmaier of c-rex.net GmbH for the continuous technical support and system access.

7 References

1. W. Ziegler: Extending intelligent content delivery in technical communication by semantics: microdocuments and content services. ETLTC2020. Aizuwakamatsu, Japan. (2020)
2. K. Alexander: Kompendium der visuellen Information und Kommunikation. Berlin Heidelberg: Springer Vieweg. (2013)
3. J. Jacobsen, L. Mayer: Praxisbuch Usability und UX. Bonn: Rheinwerk Verlag (2017)
4. B. Goldstein: Wahrnehmungspsychologie. Der Grundkurs. Berlin Heidelberg: Springer Verlag (2015)
5. U. Parson, J. Sapara, W. Ziegler: iiRDS for Technical Writers - Introduction to the Metadata. Tagungsband zur tekom Jahrestagung (2017)
6. c-rex.net, Digital Transformation Service, https://c-rex.net/
7. R. Fielding: Architectural Styles and the Design of Network-based Software Architectures. Irvine, CA: University of California (2000)
8. J. Grossmann: Introduction to Model/View/ViewModel pattern for building WPF apps. Available online at: https://docs.microsoft.com/en-us/archive/blogs/johngossman/introduction-to-modelviewviewmodel-pattern-for-building-wpf-apps, checked 3/16/2020. (2020)
9. amCharts, Programming library for data visualization, https://www.amcharts.com/
10. BootstrapVue, Vue extension, https://bootstrap-vue.org/